3-9 パッチの構造化

Maxでは、手順が簡単なこともあって、思いつくままにパッチを拡張していくことができる。しかし、その反面、しばしば複雑に入り組んだ巨大なパッチを作りがちだ。そのようなパッチをあとで見直すと、自分で作ったものであっても、どのような処理が行われているか理解しにくい。しかも、間違いの発見や将来の改造などが困難になる。そこで、パッチをまとまりごとに構造化することで、プログラミングの効率化を図ることができる。ここでは、そのような手法を説明する。

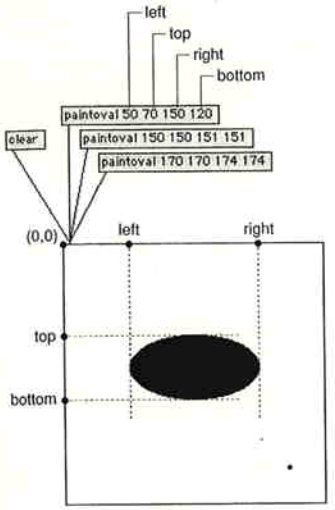
ドットを描く処理

ここでの題材として、ビットマップ描画を行うlcdオブジェクトの任意の位置にドット(点)を打つパッチを考えよう。これはグラフを描きたい場合や、点描画のような絵を描きたいときに必要になる。lcdオブジェクトには、さまざまな種類の描画を行うメッセージがあるが、点を打つメッセージはない。また、1ピクセルの点だけでなく、ある程度大きさを持ったドット(小円)も描けるようにしたい。このような描画のためには、内部を塗りつぶした円を描くpaintovalメッセージを利用するのがよいだろう。

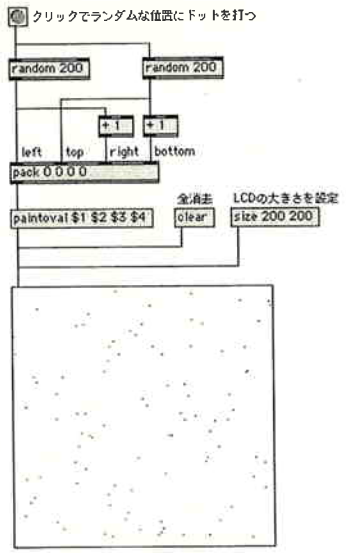
まず、1ピクセルのドットを描くことを考えてみよう。paintovalメッセージは、4つの整数アーギュメントを持ち、順に、円の左端座標(left)、上端座標(top)、右端座標(right)、下端座標(bottom)を表す。3-9-1のパッチでは、メッセージ・ボックスをクリックすれば、楕円形、1ピクセルの点、4ピクセルの小円が描かれる。ちなみに、clearメッセージはlcdオブジェクトの全領域を消去する。

次にbuttonオブジェクトをクリックするごとに、ランダムな位置にドットを打つようにパッチを作ってみたのが3-9-2のパッチである。まず、インスペクターを使ってlcdオブジェクトの大きさを縦横ともに200ピクセルに設定する。これは、lcdオブジェクトにsize 200 200というメッセージを送ることでも可能だ。ランダムな左端座標はrandom 200によって求めることができる。これに1を加えたものが右端座標となる。同様に、上端座標をrandom 200によって求め、これに1を加えると下端座標になる。これらの数値を受け取ったunpack 0 0 0 0オブジェクトは、4つの数値からなるリストを出力する(なお、実

■3-9-1 paintovalメッセージとアーギュメント



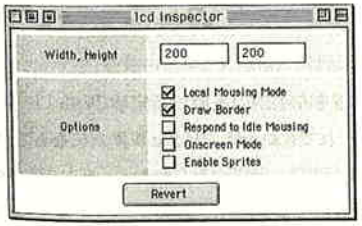
■3-9-2 ランダムな位置にドットを打つ



際にはright-to-left orderのルールに基づいて、右側のオブジェクトから処理される)。

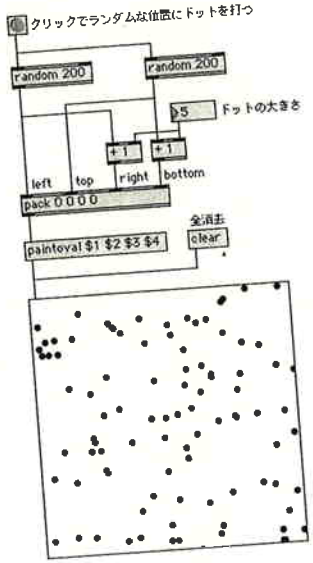
このリストはpaintoval $1 $2 $3 $4メッセージ・ボックスによって、4つの数値を伴ったpaintovalメッセージになり、lcdオブジェクトに送られる。これで、buttonオブジェクトをクリックするごとに、ランダムな位置にドットが描かれるわけだ。

■3-9-3 lcdオブジェクトのインスペクター

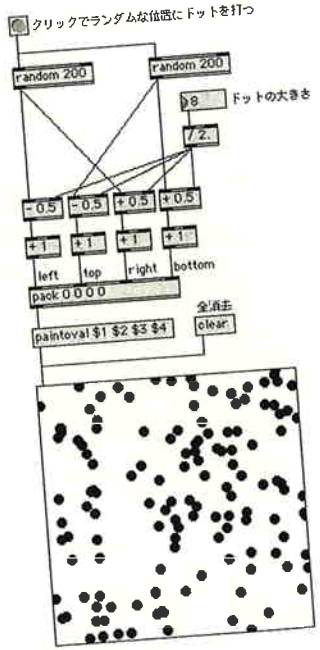


次は、ドットの大きさを指定できるようにしよう。これまでは、+1オブジェクトによって円の大きさが1ピクセルになっていたので、+1オブジェクトの第2インレットに整数を入力すれば、任意の大きさに設定することができる。そこで、ナンバー・ボックスを作り、それぞれの+1オブジェクトの第2インレットにつなぐ。これで、ナンバー・ボックスで指定した大きさのドットが描かれるようになる。

■3-9-4 指定した大きさのドットを打つ



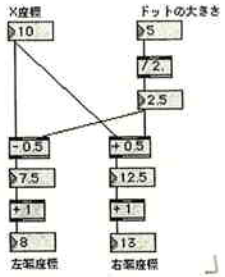
■3-9-5 ドットの位置を中心座標で指定するように改良



ところで、このパッチではドットの位置を左上隅の座標で指定している。ドットの大きさが1ピクセルや2ピクセルの場合はそれでもよいが、3ピクセル以上になることを考慮すれば、位置はドット(小円)の中心座標として指定する方がよい。そこでドットの位置を中心座標で指定するように改良すれば、3-9-5のようになる。

ここでは、ドットの大きさを2で割った値を中心座標から引いたり、足したりすることで左、上、右、下の各座標を得ている。この計算は、点の大きさが奇数の場合があるので、実数として計算しなければならない。また、座標は整数で指定するが、実数から整数に変換する際に、小数点以下が切り捨てられるので、1を加えている。このような処理は、次のような確認用のパッチを作って確かめるとよい。

■3-9-6 処理確認用のパッチ



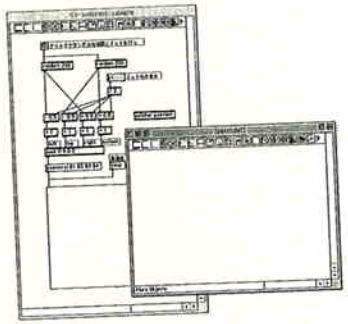
patcherオブジェクトによるサブ・パッチ

このようにして、ドットを打つパッチを作ることができたが、随分とパッチが複雑になってしまった。一方、ドットを打つという目的からすれば、例えば、中心座標(100,50)に大きさ3のドットを描く場合は、paintdot 100 50 3といったメッセージを用いたいところだ。残念ながら、既存のオブジェクトに新しいメッセージを追加することはできないが、新しいオブジェクトを作ることができるようになっている。

そこで、中心座標と大きさを入力すると、座標計算を行ってpaintovalメッセージを出力するオブジェクトを作ってみよう。Maxではpatcherというオブジェクトによって、新しいオブジェクトを作ることができる。実際には、patcherオブジェクトの中身はパッチであり、サブ・パッチとも呼ばれる。patcherオブジェクトにはアーギュメントとしてサブ・パッチの名前を与える。サブ・パッチ名は省略しても構わないが、混乱しないように適切な名前を付ける方がよいだろう。また、patcherはpと省略することができる。

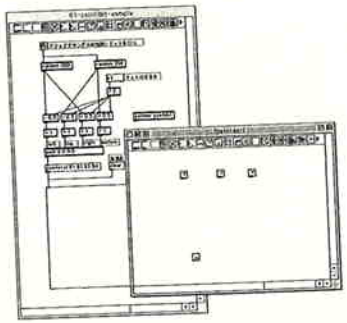
では、新しくオブジェクト・ボックスを作り、patcher paintdotとタイプしてほしい。paintdotでもよい。そしてenterキーを押すと、新しいパッチ・ウィンドウが開き、タイトル・バーには[paintdot]と表示される。このウィンドウ内に、paintdotサブ・パッチの中身となるパッチを作ることになる。

■3-9-7 patcherオブジェクトを使ってpaintdotサブ・パッチを作成



このサブ・パッチは、入力として中心点のX座標、Y座標、そして大きさの3つの数値を受け取り、paintovalメッセージを出力することになる。したがって、入力が3つ、出力が1つなので、inlet《Inlet》オブジェクトを3つ、outlet《Outlet》オブジェクトを1つ作成する。inletオブジェクトはオブジェクト・パレットの9番目、outletオブジェクトはオブジェクト・パレットの10番目にある。形が似ているので、間違わないようにしよう。

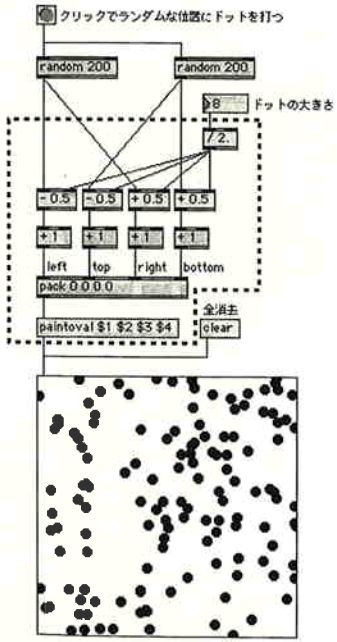
■3-9-8 inletオブジェクトとoutletオブジェクトを作成



inletオブジェクトとoutletオブジェクトを作成すると、自動的にpatcher paintdotオブジェクトにも同じ数のインレットとアウトレットが作られる。これらは互いに結びついており、patcher paintdotオブジェクトの1番目のインレットに入力されたメッセージは、paintdotサブ・パッチの一番左のinletオブジェクトから出力される。そして、2番目のインレットは2番目のinletオブジェクト、3番目のインレットは3番目のinletオブジェクトに対応している。また、paintdotサブ・パッチのoutletオブジェクトにメッセージを送ると、patcher paintdotオブジェクトのアウトレットから出力される。

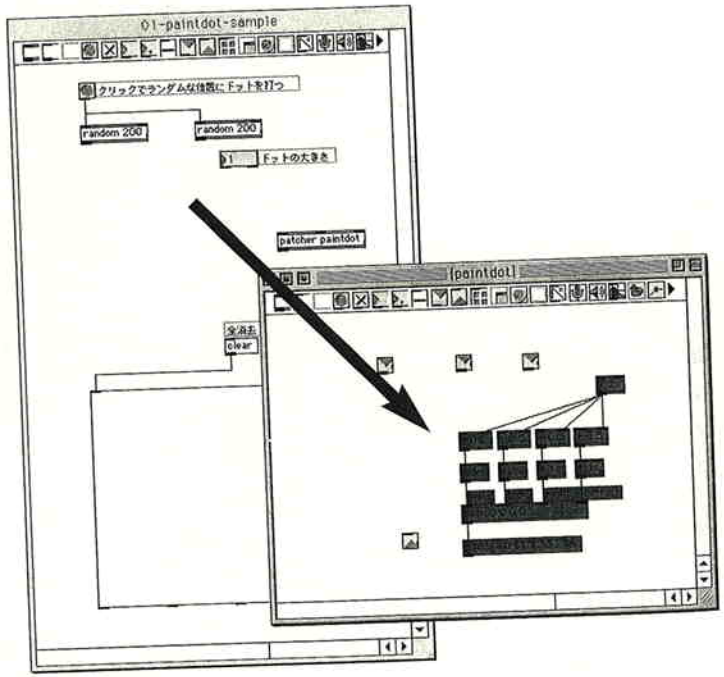
さて、元の3-9-5のパッチで考えると、次の3-9-9の点線で囲った部分を、paintdotサブ・パッチの処理として行えばよいことになる。

■3-9-9 元のパッチでサブ・パッチ化する部分



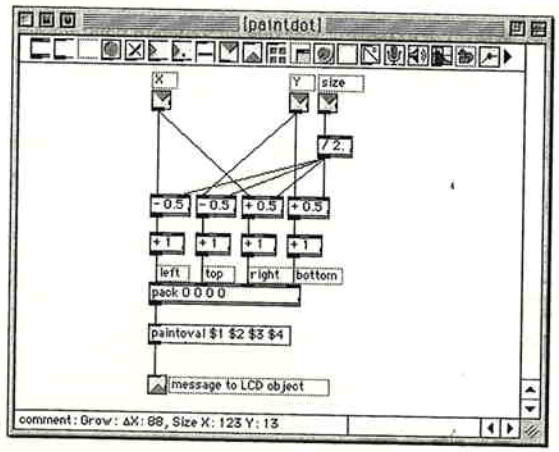
そこで、これらのオブジェクトを選択し、EditメニューからCutを選ぶ。そして、paintdotサブ・パッチを開き、EditメニューからPasteを選ぶ。これで、必要な部分が元となるパッチからサブ・パッチに移動する。

■3-9-10 元のパッチからサブ・パッチへ必要な部分を移動



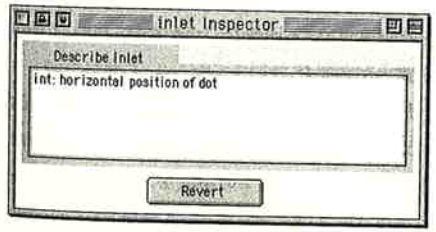
次に、paintdotサブ・パッチのinletオブジェクトやoutletオブジェクトと、移動したオブジェクトとをパッチ・コードでつなぐ。インレットやアウトレットとの対応関係に応じて、間違わないように接続したい。

■3-9-11 paintdotサブ・パッチ内のパッチ・コードを接続



また、混乱や誤解を避けるために各オブジェクトに適切なコメントを付けておく方がよい。inletオブジェクトやoutletオブジェクトを選択し、EditメニューからGet Info...を選んでインスペクターを開いて入出力するメッセージの種類や役割などを簡潔に記述する。インスペクターで記入した説明文はpatcher paintdotオブジェクトのインレットやアウトレットにマウス・ポインターを近づけたときに、アシスタンス・エリアに表示される。

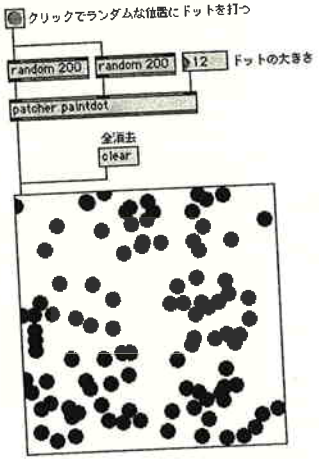
■3-9-12 inletオブジェクトのインスペクター



作成作業が終了したら、paintdotサブ・パッチは閉じても構わない。再度、修正が必要になれば、patcher paintdotオブジェクトをダブル・クリックすることで、paintdotサブ・パッチが開く。最後に、元のパッチ上で今作成したpatcher paintdotオブジェクトを他のオブジェクトにつなげばよい。

これで、ドットを描く処理をサブ・パッチ化することができた。patcher paintdotオブジェクトのインレットやアウトレットのアシスタンスについても確かめておこう。

■3-9-13 サブ・パッチを用いて整理した3-9-5のパッチ



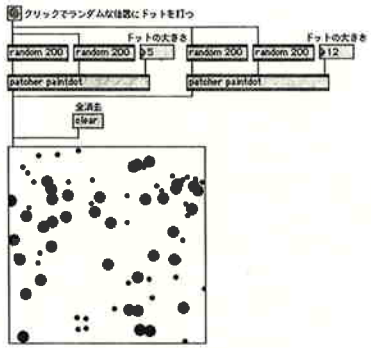
このようにサブ・パッチを利用すれば、すっきりとしたパッチを作ることができる。ある程度規模が大きくなったパッチは、意味のある処理のまとまりごとにサブ・パッチ化するのがよいだろう。慣れてくれば、最初からサブ・パッチを作りながら、全体のパッチを作ってもよい。もちろん、処理内容が連想できる明快なパッチ名や、使いやすいインレットとアウトレットの構成を考えることが必要だ。

また、ひとまとまりの処理をサブ・パッチ化することは、細かな処理内容を元のパッチから隠蔽していることになる。カプセル化と言ってもよいだろう。1つのパッチに、いくつでもサブ・パッチを含めることができ、サブ・パッチの中にサブ・パッチを持たせることもできる。このようなメカニズムを使って、パッチを階層化して整理するとよい。

❷ サブ・パッチの複製

patcherオブジェクトを使ったサブ・パッチは、他のプログラミング言語でのサブ・ルーチンや関数に似ている。あるサブ・パッチをコピーしてペーストすれば、複数のサブ・パッチを同時に使うことや、他のパッチで利用することができる。次の例では、patcher paintdotオブジェクトの複製を作り、異なる大きさを入力して、2種類のドットを同時に描いている。

■3-9-14 複製したpatcher paintdotオブジェクト



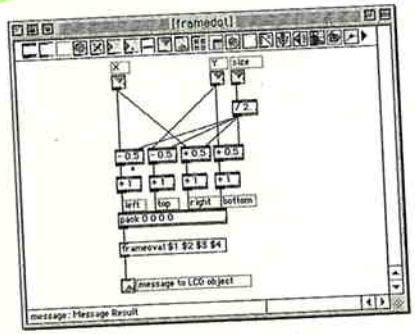
このように、あるサブ・パッチの複製を作ることで、同じ処理を行うことができる。しかし、実際には、それぞれのサブ・パッチは独立した異なるオブジェクトになっており、サブ・パッチの中身が同じなので、同じ処理を行うに過ぎない。そのため、どちらかのサブ・パッチの内容を変更すれば、異なる処理を行うようになる。

また、新しいオブジェクト・ボックスを作って、patcher paintdotと入力しても、その中身は空であり、既存のpatcher paintdotオブジェクトを反映するわけではない。この点で、サブ・パッチはサブ・ルーチンや関数とは異なると考えるべきだ。

しかし、一部は異なるが同じような処理を行いたい場合は、サブ・パッチの複製が役に立つだろう。サブ・パッチを複製したあと、いずれかのサブ・パッチの内容を変更すればよい。先のパッチを発展させて、一方のサブ・パッチはpaintovalメッセージではなくframeovalメッセージを出力するように変更してみよう。frameovalはlcdオブジェ

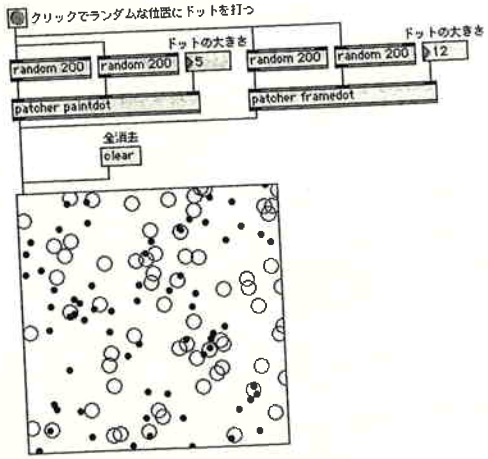
クトに縁だけの円を描くメッセージだ。具体的な変更箇所としては、サブ・パッチ内のpaintoval $1 $2 $3 $4というメッセージ・ボックスをframeoval $1 $2 $3 $4のように変えるだけでよい(なお、作例ではサブ・パッチ名も合わせてframedotと変更している)。

■3-9-15 内容を変更したframedotサブ・パッチ



以上で塗りつぶしたドットと縁だけのドットを描くパッチの完成だ。framedotに相当するパッチを一から作るよりも、はるかに手間を簡略化できることが分かるだろう。

■3-9-16 複製したpatcher paintdotオブジェクト

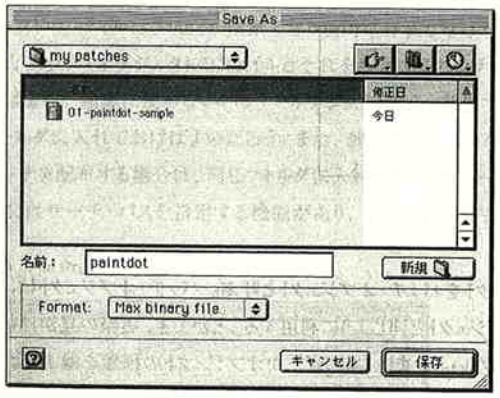


2006/5/2

パッチ・オブジェクトの作成

patcherオブジェクトによるサブ・パッチの複製は独立した存在であり、それぞれ内容を変更できる。一方、まったく同じサブ・パッチを何度も使いたい場合があるだろう。このためには、サブ・パッチをファイルとして保存すればよい。先の例では、patcher paintdotオブジェクトをダブル・クリックしてサブ・パッチのウィンドウを開き、Fileメニューから SaveまたはSave As...を選ぶ。これでファイル保存ダイアログが開くので、適切なファイル名を設定し、適当な場所を選んで保存する。ここでは、ファイル名はサブ・パッチ名と同じpaintdotとし、元のパッチ・ファイルと同じ場所に保存することにしよう。ファイル形式はバイナリーでもテキストでも構わない。

■3-9-17 サブ・パッチのファイル保存



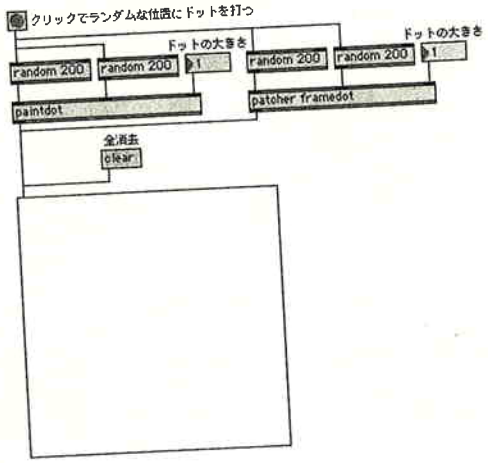
次に、パッチ・ウィンドウに新しいオブジェクト・ボックスを作り、paintdotとタイプしてenterキーを押す。これでpaintdotオブジェクトが作成される。

■3-9-18 paintdotオブジェクト

3-9-19 作成手順
① patcher paintdot を
paintdot オブジェクトへ入替えるだけ

このpaintdotオブジェクトには、3つのインレットと1つのアウトレットがあり、paintdotサブ・パッチと同じように使うことができる。

■3-9-19 paintdotオブジェクトを用いたパッチ



このようなオブジェクトをパッチ・オブジェクトと呼ぶ。パッチ・オブジェクトは、Maxに用意された標準オブジェクトと同じように利用することができ、実際の見かけも標準のオブジェクトと変わらない。つまり、ユーザーがオブジェクトの種類を増すことができ、Maxの機能を拡張することができる。正しく動作するパッチ・オブジェクトを作ることができれば、以後はその役割やインレットとアウトレットの意味を知っているだけでよく、パッチ・オブジェクトをブラック・ボックスとして用いることができるわけだ。

なお、ここでは、サブ・パッチからパッチ・オブジェクトのファイルを作成したが、新しいパッチ・ウィンドウを開いて、パッチ・オブジェクトを直接作成しても構わない。

ちなみに、パッチ・オブジェクトをダブル・クリックすると、パッチ・ウィンドウが開いて中身を確認することができる。しかし、パッチをアンロック状態にすることはできず、その内容の編集はできない。パッチ・オブジェクトの内容を変更するには、そのパッチ・ファイルを開いて編集する必要がある。パッチを編集し、ファイルを保存すれば、すでに作ったパッチ・オブジェクトにも変更内容が反映される。複数の同一パッチ・オブジェクトを利用している場合にも、すべての内容を一括して変更できるので便利である。

サーチ・パス

patcherオブジェクトによるサブ・パッチとは違い、パッチ・ファイルによるパッチ・オブジェクトは一般的なプログラミング言語でのサブ・ルーチンや関数と考えてよいだろう。オブジェクト指向プログラミングの概念で言えば、パッチ・ファイルを作成することはクラスを記述していることに相当し、パッチ・オブジェクトを作成することはインスタンスを生成することに相当する。

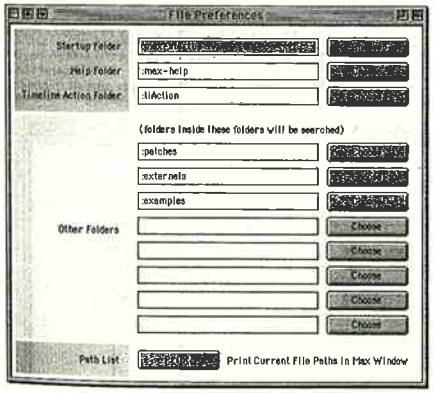
05/1/5
05/2/14 再

● サーチ・パスの設定

作成したパッチ・オブジェクトは、他のパッチでも利用できる。ただし、そのパッチ・オブジェクトのファイルは、Maxが検索できる場所に存在しなければならない。これには、まず、パッチ・オブジェクトのファイルとそれを利用するパッチ・ファイルが同じ場所にあればよい。頻繁に使わないパッチ・オブジェクトなら、利用するパッチ・ファイルと同じフォルダに入れておけばよいだろう。また、独自に作成したパッチ・オブジェクトを含むパッチを配布する場合は、同じフォルダに入れておくべきだ。

Maxにはサーチ・パスを指定する機能があり、サーチ・パス内にあるファイルを自動

■3-9-20 File Preferencesウィンドウ

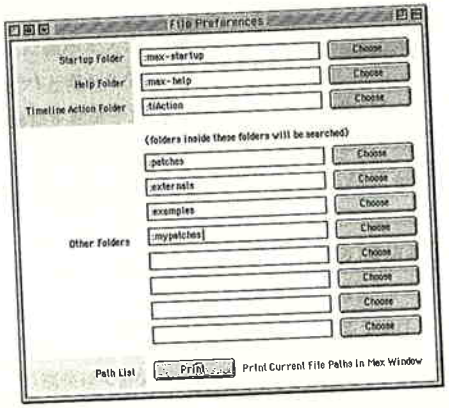


的に探し出してくれる。サーチ・パスはOptionsメニューのFile Preferences...を選び、File Preferencesウィンドウで設定する。この中のOther Foldersのテキスト・フィールドにサーチ・パスを入力すればよい。

あるいは、テキスト・フィールドの右側にあるChooseボタンをクリックすれば、フォルダ選択ダイアログが開くので、指定したいフォルダを選択して、選択ボタンをクリックすれば、そのフォルダのパスがテキスト・フィールドに設定される。

■3-9-21 フォルダ選択ダイアログ

■3-9-22 追加されたファイル検索パス



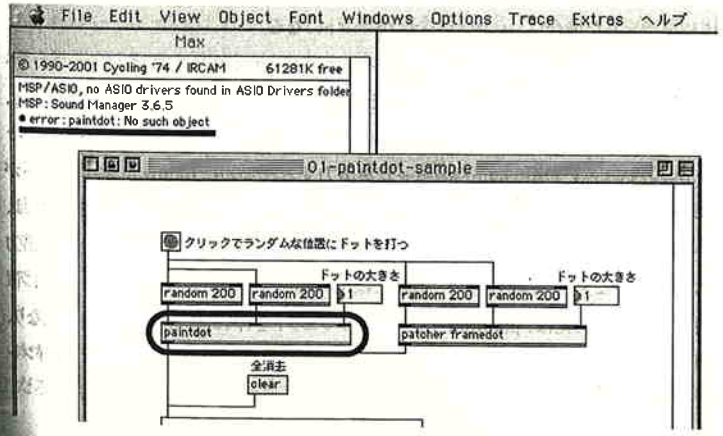
Max/MSPアプリケーションからの相対的なサーチ・パスは：(コロン)に続けてフォルダ名を表記する。サブ・フォルダは：で区切って表す。例えば :mypatchesならMax/MSPアプリケーションが含まれるフォルダにあるmypatchesフォルダを指し、:mypatches:specialならmypatchesフォルダ内のspecialフォルダを指す。フォルダ名の最後に：を付ければ、そのサブ・フォルダもすべて検索対象となる。

また、絶対的なサーチ・パスとしては、：を付けずにボリューム名から始めて、フォルダ名を：で区切って表す。例えばMyDisk:Music:Stocksなら、MyDiskというボリュームのMusicフォルダ内にあるStocksフォルダを指すことになる。

検索パスとして指定するフォルダ名にスペース(空白文字)が含まれている場合は、パスを表す文字列全体を‘と’で囲む必要がある。また、日本語などの2バイト文字は文字化けしてしまう。そのため、なるべくスペースを含まず、半角の英数字でフォルダの名前を付ける方が無難だ。

ちなみに、Other Foldersには、最初から:patches、:externals、:examplesの3つのフォルダが設定されている。これらはMax標準のオブジェクト・ファイルやパッチ・ファイルが納められているフォルダである。これらのフォルダに自分で作成したパッチ・オブジェクトのファイルを入れても構わない。しかし、混乱を避けるためには、独自のフォルダを作成してOther Foldersで指定するのがよいだろう。

■3-9-23 オブジェクトが見つからない場合の表示



ところで、サーチ・パス内でオブジェクト・ファイルが見つからなかったときは、オブジェクトの上下が二重線で描かれるところが点線で描かれ、Maxウィンドウにはオブジェクト名とともにNo such objectというエラー・メッセージが表示される。このような場合は、サーチ・パスの設定やオブジェクト・ファイルの存在を確認する必要がある。

なお、サーチ・パスによって検索されるファイルはオブジェクト・ファイルに限らない。テキスト・ファイルやサウンド・ファイルなど、あらゆる種類のファイルが対象になる。Maxが何らかのファイルを読み込む必要があるときは、サーチ・パスに従ってファイルの検索を行うことになる。また、あるパッチから何らかのファイルを読み込む際には、File Preferencesウィンドウで設定したサーチ・パスよりも、そのパッチ・ファイルがあるフォルダ内が優先される。

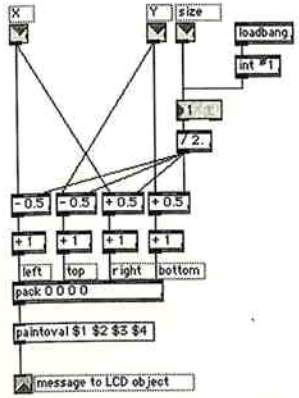
パッチ・オブジェクトのアーギュメント

パッチ・オブジェクトにもアーギュメントを指定することができる。ここではpaintdot 12のようにして、アーギュメントとしてドットの大きさを指定できるようにパッチを改良しよう。

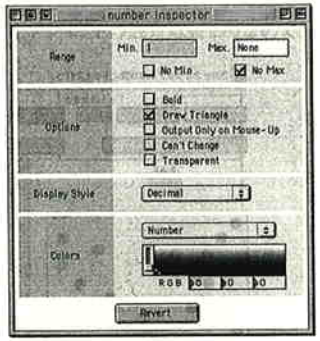
パッチ・オブジェクトのパッチでは、そのアーギュメントを#1や#2のように#記号とアーギュメント番号を示す整数で参照する。この#1などは、パッチ内のオブジェクトのアーギュメントとして用い、パッチ・オブジェクトが作られるとアーギュメントに与えた数値やシンボルに置き換えられる。例えば、+ #1というオブジェクトがあり、アーギュメントに12を指定した場合は、+ 12というオブジェクトになる。

先のpaintdotオブジェクトでは、3-9-24のようにパッチを改良すれば、ドットの大きさを指定するアーギュメントを受け付けることができる。新たに加わったのは、loadbang、int #1、そしてナンバー・ボックスのオブジェクトだ。まず、loadbangオブジェクトはパッチが読み込まれたときにbangメッセージを出力する。つまり、paintdotオブジェクトが作られた際に、bangメッセージを出力することになる。次のint #1のintオブジェクトは、アーギュメントで指定した整数を蓄え、bangメッセージを受け取ると、その整数を出力する。アーギュメントは#1なので、実際にはpaintdotオブジェクトのアーギュメントに置き換えられる。例えば、paintdot 12というオブジェクトを作れば、int #1はint 12となり、loadbangオブジェクトからbangが送られるので、12を出力する。ここからは第3インレットに数値が入力された場合と同じ処理となる。

■3-9-24 アーギュメントを指定できるpaintdotパッチ



■3-9-25 ナンバー・ボックスの設定



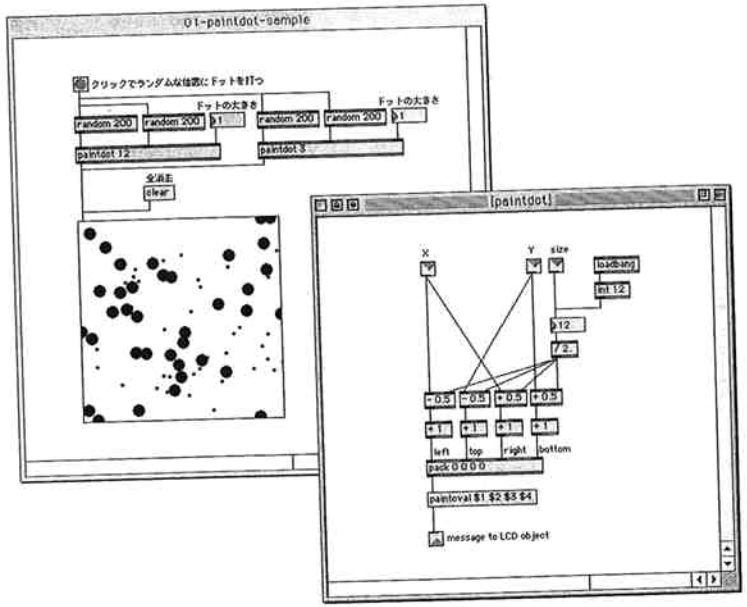
ここで、ドットの大きさを0または負の数値とした場合は、エラーにはならないものの、実際の描画が行われない。特に、アーギュメントを意識せずにpaintdotというオブジェクトを作った場合に、#1の省略値として0が用いられるので問題になるだろう。

このような場合の対策として、/ 2.オブジェクトの前にナンバー・ボックスをつないでいる。ナンバー・ボックスはインスペクターで最小値を1に設定している。したがって、0以下の数値がナンバー・ボックスに入力された場合は1が出力されることになる。これで、アーギュメントの指定だけでなく、第3インレットに0以下の数値が入力された場合も、ドットの大きさは1となり、点が描かれる。

paintdotオブジェクトの修正後パッチを保存し、改めてpaintdot 12のようにアーギュメントを指定してみよう。これでアーギュメントに指定した大きさのドットが描かれるはずである。もちろん、第3インレットに数値を入力することで、ドットの大きさを変えることもできる。

なお、paintdot 12オブジェクトをダブル・クリックすれば、そのパッチ・ウィンドウが開かれるが、int #1がint 12のようにアーギュメントの値に置き換わっているのが分かるだろう。paintdot 3ならint 3となっているはずだ。

■3-9-26 アーギュメントを指定したpaintdotオブジェクトとその中身



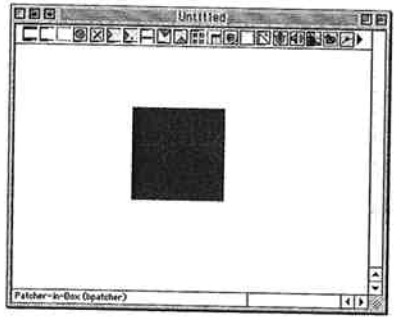
bpatcherオブジェクトによるパッチ表示

パッチ・オブジェクトとしてパッチを利用する以外に、bpatcher《Patch-in-Box》によってパッチを利用する方法もある。bpatcherオブジェクトは、ちょうど覗き窓のようにオブジェクト内にパッチそのものを表示する。bpatcherオブジェクトはオブジェクト・パレットの12番目にあるので、これをパッチ・ウィンドウに作成しよう。点線で描かれた矩形がbpatcherオブジェクトだ。

次に、bpatcherオブジェクトのインスペクターを開き、Patcher File欄のOpen...ボタンをクリックする。ファイル選択ダイアログが開いたら、bpatcherオブジェクトで利用するパッチ・ファイルを選択。ここではpaintdotファイルを選ぼう。これでPatcher File欄にパッチのファイル名が表示される。

また、Arguments to Patcher欄は、パッチへのアーギュメントの指定なので、ここではドットの大きさとして12としよう。これでbpatcherオブジェクトの設定は終わりなので、インスペクターを閉じる。

■3-9-27 bpatcherオブジェクトの作成

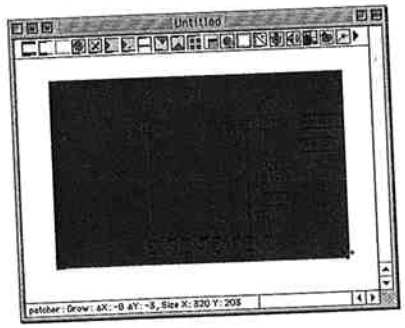


■3-9-28 bpatcherオブジェクトのインスペクター

このようにして、bpatcherオブジェクトで利用するパッチを指定すれば、bpatcherオブジェクトに3つのインレットと1つのアウトレットが作られる。また、bpatcherオブジェク

ト内には何も表示されないかもしれないが、オブジェクトを大きくすると、paintdotパッチが見えてくるだろう。つまり、bpatcherオブジェクトはパッチの一部を表示しながら、パッチを利用することができるわけだ。

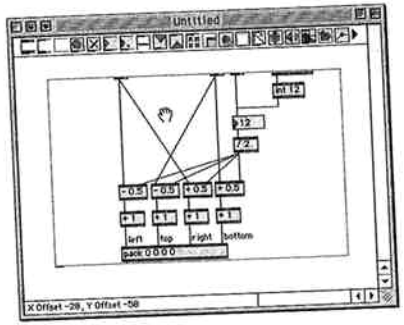
■3-9-29　bpatcherオブジェクトでのパッチ表示



windowsでは Ctrlキー

また、commandキーとshiftキーを押しながらbpatcherオブジェクトをドラッグすると、マウス・ポインターが手の形になり、パッチをずらして表示する部分を変えることができる。このようにして、表示させたい部分を設定すればよい。また、offsetメッセージによって、パッチのずらし方を設定することもできる。

■3-9-30　パッチの表示部分をずらす



bpatcherオブジェクトが役立つのは、利用するパッチがユーザー・インターフェース・オブジェクトを多く含んでいる場合だろう。そのようなパッチでは、bpatcherオブジェクトを使って処理機能だけでなくユーザー・インターフェース・オブジェクトも再利用できるからだ。そこで、ランダムな位置にドットを描くパッチをbpatcherオブジェクトを利用するように改良してみよう。もっとも、paintdotオブジェクトで使用しているユーザー・インターフェース・オブジェクトは、ドットの大きさを設定するナンバー・ボックスだけなので、bpatcherオブジェクトの有難味は少ないかもしれない。

ではまずbpatcherオブジェクトで表示する必要がないオブジェクトやパッチ・コードを隠そう。手順はpaintdotのパッチを開いたら、EditメニューのSelect Allを選んですべてを選択し、ObjectメニューのHide on Lockを選ぶ。これでロック時には何も表示されなくなる。次にナンバー・ボックスとcommentオブジェクトを選択し、ObjectメニューからShow on Lockを選ぶ。これで、ロック時にはナンバー・ボックスとcommentオブジェクトだけが表示されるようになる。このように設定をしたら、パッチを保存して閉じる。

■3-9-31　ナンバー・ボックスとcommentオブジェクトだけをロック時に表示するように設定